a task module, selectively communicating with each of said plurality of event modules and said subset of said plurality of processing modules, said task module including code for executing selected ones of said plurality of scripts that corresponds to said first and second event data signals, for executing said plurality of instructions within said selected scripts, and for selectively invoking one of said plurality of processing modules in accordance with information regarding statuses of said plurality of processing modules and results generated by previously executed instructions;

wherein during execution of said selected scripts, each of said selected scripts incorporates said processed data to modify execution of next instructions of said selected scripts. --

REMARKS

In the present Office Action claims 1-46 were examined. Claims 1-46 were rejected, and no claims were allowed.

By this Amendment, claims 1-6, 8-20, 24, 27-34, 42 and 45 have been amended, no claims were cancelled and claims 60-63 have been added. Accordingly, claims 1-46 and 60-63 are presented for further examination.

In Section 3 of the Office Action the Examiner rejects claims 1-4, 20-27, 34-35, 41-44 and 46 under 35 U.S.C. 103(a) as being unpatentable over Krishnamurthy et al. (a previously cited technical journal article) in view of Mahajan (U.S. Patent No. 5,404,528). This rejection is respectfully disagreed with, and is traversed below.

The arguments made previously are repeated and incorporated by reference herein and, in particular, the discussions of the disclosure of Krishnamurthy et al. wherein it is submitted that Krishnamurthy et al. execute commands in a static and predefined manner.

In apparent agreement with applicants' previous arguments, the Examiner highlights some of the deficiencies of Krishnamurthy et al. (from the last paragraph of page 2 to the first full paragraph of page 3 of the Office Action) where, in summary, the Examiner states that Krishnamurthy et al. do not describe, during execution of a script, (1) modifying a processing flow of the script, or (2) incorporating processed data into one or more instructions of the script. The Examiner also states that Krishnamurthy et al. do not describe (3) a task module that selectively communicates with processing modules. The Examiner attempts to cure these deficiencies by combining Krishnamurthy et al. with the description of Mahajan.

Mahajan is seen to describe a scripting system wherein predefined portions of application programs ("exported functionality") are made available for execution by a script via a prearranged entry point to the application programs stored in a DLL list (col. 4, lines 10-32). A script is invoked upon the occurrence of a triggering event. Mahajan describes that triggering events are signaled to a script interpreter by an application program. In response to the triggering event, the script interpreter references a static script/trigger event correlation list, retrieves a script associated to the triggering event in the correlation list and interprets the script. As a statement is encountered within the script that includes a request for exported functionality, the script interpreter references the DLL list and executes the exported functionality. When the exported functionality completes control is passed back to the script interpreter, which continues interpreting the script at the next statement in the script until all statements are completed (col. 5, lines 3-23). As noted by the Examiner, the exported functionality may return data to the invoking script. The invoking script may then evaluate the returned data during subsequent processing, e.g., the "IF...END" logic highlighted by the Examiner.

Accordingly, Mahajan is seen to describe a scripting system in which a script invoked in response to a triggering event may individually execute predefined, external functionality made available to the script in advance by other application programs (e.g., the statements requesting exported functionality made available though predefined, static correlation and DLL lists). In effect, Mahajan is seen to describe a predetermined coupling (via the correlation and DLL lists) of a script, the script interpreter, and the exported functionality. As argued below, it is submitted that the proposed combination of Krishnamurthy et al. and Mahajan does not describe or suggest the subject matter of the present invention.

The present invention teaches a dynamic execution of instructions and processing modules to gather data in response to event data. Clarifying amendments have been made to independent claims 1 and 34 to even further distinguish the claimed invention from the cited documents. For example, independent claim 1 now positively recites selectively invoking processing modules in accordance with information regarding the status of the processing modules and results generated by previously executed instructions. Support for the clarifying amendments may be found in the Specification, as filed, at least at page 13, lines 3-9.

It is submitted that the proposed combination of Krishnamurthy et al. and Mahajan do not expressly or implicitly describe or suggest scripts that dynamically invoke processing in accordance with status information and process data generated by previously executed instructions, and that modify execution of next instructions by incorporating data processed by the dynamically invoked processing modules. Rather, the proposed combination merely provides for static execution of scripts that may include reference to predefined, external functionality.

In view of the foregoing, applicants submit that independent claims 1 and 34 are patentable over the proposed combination. Since independent claims 1 and 34 are deemed patentable, claims

2-4, 20-27, 35, 41-44 and 46 are deemed patentable at least for the reason that the claims depend from allowable base claims. Based thereon, the Examiner is respectfully requested to reconsider and to remove the rejection of claims 1-4, 20-27, 34-35, 41-44 and 46, as now amended, under 35 U.S.C. 103(a).

In Section 4 of the Office Action the Examiner rejects claims 5-19, 28-33, 36-40 and 45 under 35 U.S.C. 103(a) as being unpatentable over Krishnamurthy et al. in view of Mahajan as applied to claim 1, and further in view of Waclawsky et al. (U.S. Patent No. 5,493,689). This rejection is respectfully disagreed with, and is traversed below.

As previously argued, Waclawsky et al. disclose an event-driven load balancing system having a filter that identifies patterns of data to determine network activity. For example, Waclawsky et al. are seen to describe data collection activities in high speed terabit and gigabit communication networks and, in particular, a complex expert system driven architecture that employs logic trees for digitally filtering binary sequences over communication networks (Summary of the Invention at col. 3, line 50 to col. 5, line 26).

It is respectfully submitted that one skilled in the art would not look to data collection activities of expert systems operating in high speed communication networks to solve the problem addressed by the present invention, namely, dynamic event-driven script processing. Thus, it is not seen that the references can be properly combined, and any attempt to combine them can only be made in light of the Applicants' disclosure. It is well settled that the combination of elements from non-analogous sources, in a manner that reconstructs the applicants' invention only with the benefit of hindsight, is insufficient to present a prima facie case of obviousness. See, for example, In re Oetiker, 977 F.2d 1443, 1447, 24 USPQ2d 1443, 1446 (Fed. Cir. 1992).

Further, it is submitted that, absent Applicants' disclosure, there is no motivation, suggestion or incentive for one skilled in the art to combine the cited art, as is suggested by the Examiner. The Examiner merely states that "[s]ince Krishnamurthy and Waclawsky address event management, it would be obvious to combine the teachings." It is submitted that neither of the references expressly or impliedly suggest the proposed combination nor does the Examiner present a convincing line of reasoning as to why one skilled in the art would have found the claimed invention to have been obvious in light of these references.

Applicants therefore conclude that the Examiner inappropriately uses Applicants' disclosure and hindsight reconstruction to pick and chose features of the cited art so as to render the claimed invention obvious. See, for example, In re Fritch, 972 F.2d 1260, 1266, 23 USPQ2d 1780, 1784 (Fed. Cir. 1992).

Since none of the references presently of record in the present application, either singly or in combination, teach or suggest a data processing system as now claimed, Applicants claims should be allowed over the cited references.

Accordingly, the Examiner is respectfully requested to reconsider and to remove the rejection of claims 5-19, 28-33, 36-40 and 45, as now amended, under 35 U.S.C. 103(a).

As part of this response claims 60-63 have been newly added. It is respectfully submitted that, at least in view of the arguments made above, the newly added claims are patentable over the cited documents.

Applicant has made a diligent and sincere effort to place this application in condition for immediate allowance and notice to this effect is earnestly solicited. If however, a notice of allowance cannot be issued, it is respectfully requested that the undersigned attorney of record be contacted to resolve any outstanding issues.

Early and favorable action is earnestly solicited.

Respectfully submitted
Schultz, Richard K., et al

DATE: April _6_, 2001

Gregory S. Rosenblatt
Attorney for Applicants
Reg. No. 32,489

WIGGIN & DANA
One Century Tower
New Haven, Connecticut 06508
Tel. No. 203-498-4566

5KS301!.DOC\12217\1\260211.03

MARKED-UP VERSION OF CLAIMS

1.   (Twice Amended)   A data processing system stored on a computer-readable medium comprising:

[one or more] a plurality of event modules each including code that generates an event data signal representative of a particular event;

[one or more] a plurality of scripts[,] each [of said one or more scripts] having [one or more] a plurality of instructions that provide results;

[one or more] a plurality of processing modules each including code that provides processed data to said [one or more] plurality of scripts; and

a task module, selectively communicating with each of said [one or more] plurality of event modules, said task module including code for [execution of] executing a selected one of said [one or more] plurality of scripts that corresponds to said event data signal, for executing said plurality of instructions within said selected script, and [also] for selectively [communicating with said] invoking at least one [or more process] of said plurality of processing modules in accordance with information regarding statuses of said plurality of processing modules and results generated by previously executed instructions;

wherein during [said] execution of said selected [script said task module invokes one or more of said one or more processing modules that process data and transmit processed data to said task module and] script, said selected script incorporates [results of] said processed data to modify execution of next [into said one or more] instructions of said selected script.

2.   (Twice Amended) The system as claimed in claim 1 wherein said task module executes [a plurality] two or more of said [one or more] plurality of scripts substantially simultaneously.

3.   (Amended) The system as claimed in claim 2 further comprising:

a converter module, in communication with said task module, including code that maps said event data signal to [one or more] at least one of said [plurality] two or more of said [one or more] plurality of scripts upon reception of said event data signal by said task module.

4.   (Amended) The system as claimed in claim 1 wherein said [one or more] plurality of processing modules provide event data signals, representative of [a] particular [event] events, to said task module.

5.   (Amended) The system as claimed in claim 1 further comprising:

a status monitoring module, in communication with said task module, including code that provides said status information to said task module [relating to] including operating conditions of said [one or more] plurality of processing modules.

6.   (Amended) The system as claimed in claim 5 wherein said status monitoring module is in direct communication [said] with said [one or more] plurality of processing modules.

7.   The system as claimed in claim 5 wherein during said execution of said selected

script, said status monitoring module stores data associated with said selected script in an

associated memory.

8.   (Amended) The system as claimed in claim 1 further comprising:

a load balancing module, in communication with said task module, including code that

dynamically selects available ones of said [one or more] plurality of processing modules to

perform processing.

9.   (Amended) The system as claimed in claim 8 wherein said load balancing module

is in direct communication with said [one or more] plurality of processing modules.

10.   (Amended) The system as claimed in claim 1 wherein said task module interfaces

with said [one or more] plurality of processing modules for bi-directionally and substantially

simultaneously transmitting data between said [one or more] plurality of processing modules and

said task module.

11.   (Amended) The system as claimed in claim 1 further comprising:

a resource management module, in communication with said task module, including code

that dynamically assigns processing functions to said [one or more] plurality of processing

modules.

12.   (Amended) The system as claimed in claim 11 wherein said resource management module is in direct communication with said [one or more] plurality of processing modules.

13.   (Amended) The system as claimed in claim 1 further comprising:

[one or more] a plurality of initiator modules including code that provides a communication interface between an associated one of said [one or more] plurality of processing modules and said task module.

14.   (Amended) The system as claimed in claim 13 wherein each of said [one or more] plurality of initiator modules communicates with said associated one of said [one or more] plurality of processing modules regardless of native applications contained on said associated one of said [one or more] plurality of processing modules.

15.   (Amended) The system as claimed in claim 13 further comprising:

a protocol disposed between each of said [one or more] plurality of initiator modules and said task module for providing a communication interface therebetween.

16.   (Amended) The system as claimed in claim 13 further comprising:

a protocol disposed between each of said [one or more] plurality of initiator modules and said associated one of said [one or more] plurality of processing modules for providing a communication interface therebetween.

17.   (Amended) The system as claimed in claim 1 further comprising:

[one or more] a plurality of client modules including code that provides a communication interface between an associated one of said [one or more] plurality of event modules and said task module.

18.   (Amended) The system as described in claim 17 further comprising:

a protocol disposed between said task module and each of said [one or more] plurality of client modules for providing a communication interface therebetween.

19.   (Amended) The system as claimed in claim 17 further comprising:

a protocol disposed between each of said [one or more] plurality of client modules and said associated one of said [one or more] plurality of event modules for providing a communication interface therebetween.

20.   (Amended) The system as claimed in claim 1 wherein each of said [one or more] plurality of scripts is preprogrammed to iteratively update its contents.

21.   The system as claimed in claim 1 further comprising:

a storage module, in communication with said task module, for providing storage for said system.

22.   The system as claimed in claim 21 wherein said storage module comprises a computer-readable medium.

23.   The system as claimed in claim 22 wherein said computer readable medium comprises a persistent memory.

24.   (Amended) The system as claimed in claim 21 further comprising:

a script building module, in communication with said storage module, including code that creates said [one or more] plurality of scripts.

25.   The system as claimed in claim 24 wherein said script building module includes a standard language interface.

26.   The system as claimed in claim 24 wherein said script building module includes a graphical user interface.

27.   (Amended) The system as claimed in claim 24 wherein said script building module dynamically updates and modifies said [one or more] plurality of scripts.

28.   (Amended) The system as claimed in claim 1 further comprising:

a protocol for providing a communication interface between said task module and each of said [one or more] plurality of event modules.

29.   (Amended) The system as claimed in claim 1 further comprising:

a protocol for providing a communication interface between said task module and each of said [one or more] plurality of processing modules.

30.   (Twice Amended) The system as claimed in claim 1 further comprising:

a responder module, in communication with said task module, including code that transmits

response data, resulting from said execution, from said task module in a particular format to said

[one or more] plurality of processing modules or in a particular format to said [one or more]

plurality of event modules.

31.   (Amended) The system as claimed in claim 1 further comprising:

an administrative module, in communication with said task module, including code that

receives and presents data that relates to said [one or more] plurality of processing modules.

32.   (Amended) The system as claimed in claim 1 further comprising:

[one or more] a plurality of application peripherals in communication with an associated

one of said [one or more] plurality of processing modules or an associated one of said [one or

more] plurality of event modules.

33.   (Twice Amended) A data processing system stored on a computer readable-

medium comprising:

[one or more] a plurality of event modules each including code that generates an event data

signal representative of a particular event;

[one or more] a plurality of scripts each [of said one or more scripts] having [one or more]

a plurality of instructions;

[one or more] a plurality of processing modules each including code that provides said [one or more] plurality of scripts with processed data;

a task module, selectively communicating with each of said [one or more] plurality of event modules, including code for [execution of] executing a selected one of said [one or more] plurality of scripts that correspond to said event data signal, for executing said plurality of instructions within said selected script, and for selectively invoking at least one of said plurality of process modules in accordance with information regarding a status of said plurality of processing modules and results generated by previously executed instructions;

wherein during [said] execution of said selected script [said task module invokes one or more of said one or more processing modules that process data and transmit processed data to said task module and] said selected script incorporates [results of] said processed data to modify execution of next [into said one or more] instructions of said selected script;

a resource management module in direct communication with said [one or more] plurality of event modules, said task module and said [one or more] plurality of processing modules, including code that dynamically assigns processing functions to said [one or more] plurality of processing modules; and

an administrative module in direct communication with said task module, said [one or more] plurality of event modules, said [one or more] plurality of processing modules and said resource management module, including code that receives and presents data relating to said [one or more] plurality of processing modules.

34.   (Twice Amended) A method of data processing comprising the steps of:

generating at least one event data signal at one or more peripheral modules;

mapping said at least one event data signal to a selected script chosen from one or more scripts, each said one or more scripts having one or more instructions for performing data gathering steps; [and]

executing said one or more instructions within said selected script; and

invoking, by a task module in accordance with status information, one or more processing modules to process data required by said selected script;

wherein during [said invoking and processing step] execution of said one or more instructions, said selected script dynamically incorporates [results of] data processed by said one or more processing modules to modify execution of next ones of [into] said one or more instructions of said selected script.

35. The method as claimed in claim 34 wherein said one or more peripheral modules and said task module communicate via a communication interface.

36. The method as claimed in claim 34 further comprising the step of:

dynamically managing operating functions of said one or more peripheral modules.

37. The method as claimed in claim 34 further comprising the steps of:

producing response data signals as a result of said executing step; and

transmitting said response data signals from said task module to selected said one or more peripheral modules.

38. The method as claimed in claim 37 further comprising the step of:

translating said response data signals transmitted from said task module into a format that said selected said one or more peripheral modules recognize.

39.   The method as claimed in claim 38 further comprising the step of:

storing said event data signals, said one or more scripts and said response data signals in a storage medium that is in communication with said task module.

40.   The method as claimed in claim 39 wherein said storage medium is persistent.

41.   The method as claimed in claim 34 further comprising the step of:

accessing a protocol to interface between said task module and selected said one or more peripheral modules.

42.   (Amended) The method as claimed in claim 34 further comprising the step of:

providing communication between said task module and each of said one or more peripheral modules such that said task module [will access] invokes only ones of said one or more peripheral modules [capable of] available for performing processing operations.

43.   The method as claimed in claim 34 wherein said executing step includes the step of:

interfacing with a plurality of said one or more peripheral modules substantially simultaneously.

44.   The method as claimed in claim 34 wherein said executing step executes a plurality of said one or more scripts substantially simultaneously.

45.   (Amended) The method as claimed in claim 34 [further comprising the step of: programming said one or more scripts such that] wherein said execution of said one or more instructions [will be executed based upon a] dynamically incorporates data gathered in previously executed [instruction] instructions.

46.   The method as claimed in claim 34 further comprising the step of:

providing results of said executing step to an administrative module for presenting information relating to said one or more peripheral modules.

Please add the following new claims.

--   60.   In a data processing system, a method for responding to event data, comprising:

receiving event data from a requesting one of a plurality of event modules;

mapping the event data to one of a plurality of scripts, the plurality of scripts including instructions for responding to event data;

executing the instructions within the mapped script to generate results;

during the execution of at least one instruction, selecting and invoking one of a plurality of processing modules available for providing processed data to the at least one instruction, the selecting and invoking in accordance with information regarding a status of the processing modules and results generated by previously executed instructions, and modifying execution of

next instructions within the mapped script in accordance with the processed data generated by

the selected and invoked processing module;

building a response profile including the generated results; and

wherein when the instructions within the mapped script are completed, transmitting the

response profile to the requesting one of the plurality of event modules.

61. The method as claimed in claim 60 wherein the generated results include event

data.

62. The method as claimed in claim 60, comprising:

tracing execution of the instructions within the mapped script and processing of the

invoked processing modules; and

wherein when a processing module fails, continuing execution of the mapped script and the

processing of the invoked processing modules from a last traced instruction.

63. A data processing system stored on a computer-readable medium comprising:

a plurality of event modules each including code that generates a first event data signal

representative of a first event;

a plurality of scripts each having a plurality of instructions that provide results;

a plurality of processing modules each including code that provides processed data, a

subset of said plurality of processing modules having code that selectively generates a second

event data signal representative of a second event; and

a task module, selectively communicating with each of said plurality of event modules and said subset of said plurality of processing modules, said task module including code for executing selected ones of said plurality of scripts that corresponds to said first and second event data signals, for executing said plurality of instructions within said selected scripts, and for selectively invoking one of said plurality of processing modules in accordance with information regarding statuses of said plurality of processing modules and results generated by previously executed instructions;

wherein during execution of said selected scripts, each of said selected scripts incorporates said processed data to modify execution of next instructions of said selected scripts.--